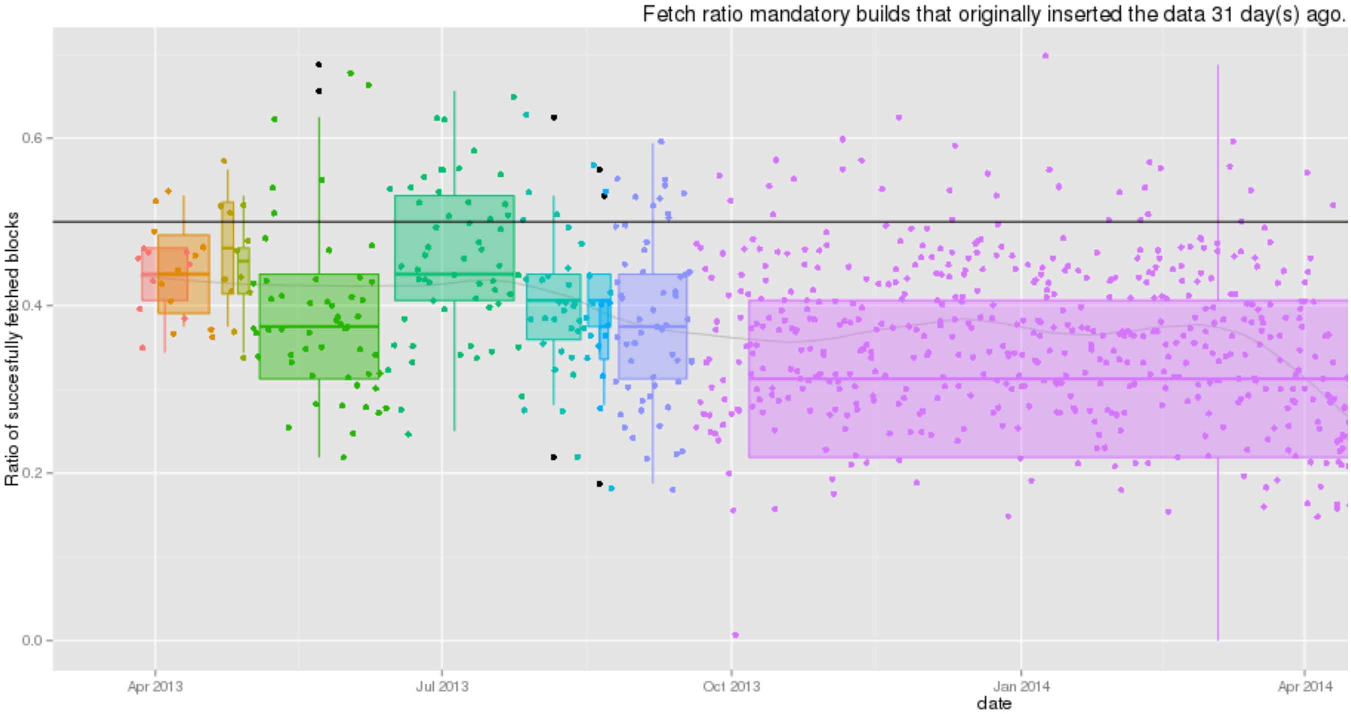
Fixing the link length distribution in Freenet

Table of Contents

this post is short. It's late and I volunteered to write it by saying "someone should". But I think it's important, so I do it. And it's interesting. It basically halted and reversed the Meltdown. Result:

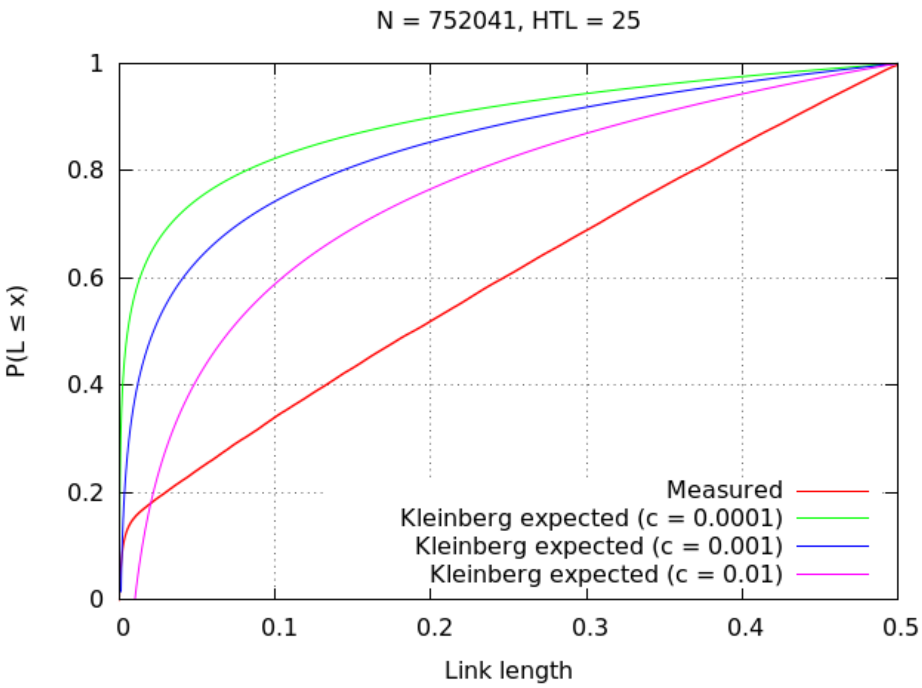


See the rectangle for 1465. Compare it with the green ones (which were before the meltdown). Below the gray line (at 0.5 = 50%) means, that files cannot be retrieved any longer. With the link length fix we are at 60%. Before the link length fix we were at roughly 45%. And this is reason to celebrate!

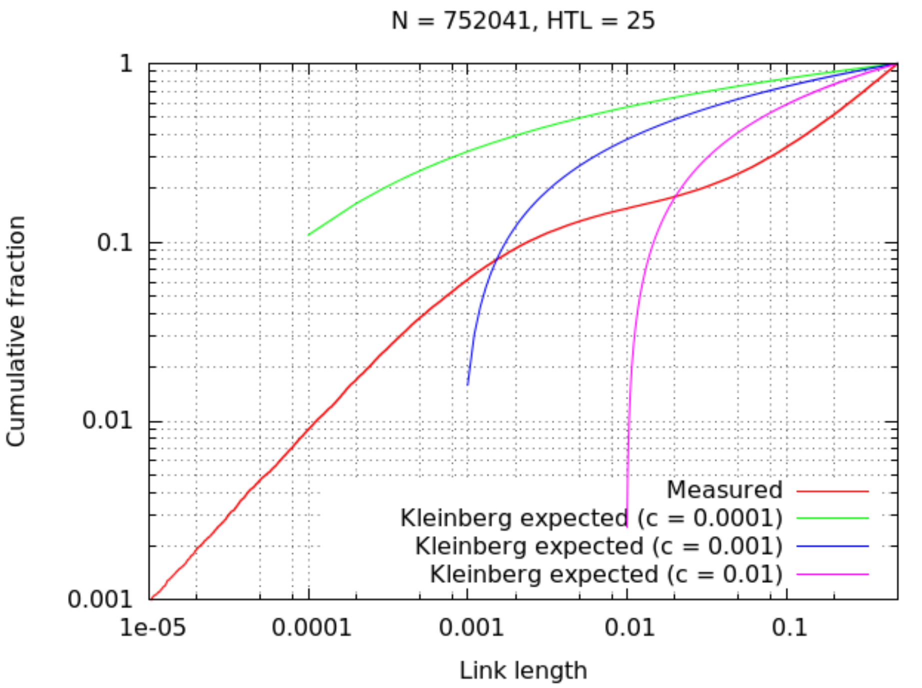
1 Motivation

For the PETS symposium 2014 there's a paper on measuring the performance of Freenet. It shows that our routing is much worse than it could be and traces that to our link length distribution: We have to many long links per node.

We've seen for years now, that our link length distribution is not ideal. Here's a graph which shows several ideal Kleinberg networks and the real freenet. As you can easily see, only around 30% of our connections are shorter than 0.1, while in a Kleinberg model 80% should be shorter than 0.1.



The paper suggest fixing the distribution by binning by link length. But looking at the graph in log-log format shows that our link length is actually really good for link lengths below 0.01.



(thanks for these graphs goes to bertm)

2 Cause

Toad easily traced this back to local requests which kill the current path folding: The locations of local requests are randomly distributed, so that long distance paths have a better chance of being successful than

short distance paths. And due to caching most content is found within 4 steps, so that local requests have a disproportionate weight for path-folding. In effect that means that nodes which are fare away quickly replace most of of our close peers and that hurts small world routing very badly. Looking at the graph it means that only 15% of our peers actually contribute to the third and later steps of small world routing while in a healthy network it should be 80%.

2.1 Why does the current routing work at all?

The one thing we wondered about for the past years was, why routing works with this structure.

Firstoff for that

2.2 Why is this broken

currently ~15% of links are <0.01. But <0.01 the distribution looks like one from Kleinberg simulations. So one where the link length distribution is roughly 1/d with d as the distance.

The assumption why that's the case is that <0.01 local requests no longer skew our link selection, so the link selection for short links is driven by requests coming from others, which ensures that our connections are a good fit for our position in the network.

3 Solution

Since the distribution of connections with a distance of less than 0.01 is good, toad is now working on limiting long-distance connections to 30% of our peers. This should massively improve routing.

As first step toad already posted a pull request for including the node location in path folding requests. operhiem1 plans to release this change this weekend as **mandatory mini-release**. This simplifies the implementation of the real fix which should make it less likely that it introduces new bugs. The node location is already easy to obtain, so this should not allow any new attacks.

Toad will then post a pull-request from which operhiem1 will decide how to go on. Likely this will first be a testing release (use `./update.sh testing`) and if no big problems turn up it should go life.

3.1 Why the 30%?

In the linear graph, you can see that the current link length distribution is linear for most of the lengths (the red line). There is a small non-linear part below a length of 0.01 which contains about 15% of the links. So about 85% of links are longer than 0.01. The green line shows that a good link length distribution would have about 80% links shorter than 0.01 - only 20% should be long. A look at the loglog graph shows that our link lengths below 0.01 exhibit nice small-world features, which they lose for longer connections.

To fit as closely as possible to the ideal network, we should only allow 20% long links. But from the practical perspective we know that the current link length distribution works. It might not be perfect, but it allows us to upload and download content decentrally. We know this from experience and we do not want to break it. Also about one third of our users has only about 12 peers. This means that if we allowed only for 20% long links, these users would only have 2 long links - links with a distribution which is known to work. So their long links would form a chain structure that would not allow for any routing decisions. In the worst case that the structure of short links should degrade massively due to some unknown effect, allowing only 20% long links could break routing completely for one third of our users. Worst case could be that we are not able to find nodes which are close enough to be counted as short link. This also illustrates, that the number of short links (or rather the definition of a short link) depends on the network size, and we would not want to encode a number which is too high, because then we would not find enough suitable peers.

30% on the other hand give even the slowest nodes at least 3 links which follow the current distribution, so routing on that graph will still be possible. Actually the number of links from the current distribution is rather 34.5%, because the current distribution includes 15% short links which are not counted against the 30%.

The third of our userbase with 20-30 connections will still have about 10 long links, so they are squarely in safe territory, even in the worst case.

And reserving 70% of the peers for short links should increase the number of actual small world significantly - up from 15%. For slow nodes that means that we no longer have only 1-2 efficiently routed peers (this is a chain structure!) with lots of random misrouting but 7 efficiently routed peers, which should give us efficient small world routing. The difference between 7 and 8 efficiently routed peers is not that high in case that reality agrees with theory, but the difference between degrading to a chain and still having 3 working links in case reality disagrees with the theory would be huge.

This low number of efficiently routed peers for slow nodes could actually explain why freenet performance can differ so much: If your node is mostly connected to others who just route, then these will already have few local requests and provide actual small world routing. If it is connected to fast nodes, then these will already have 6-15 efficiently routed peers, so you get real small world routing. If fast internet connections correllate with long uptimes, then if you're using freenet 24/7 many of your connections will be to fast nodes and as such you will see real small world routing.

3.2 Why not just limit path folding to low HTL?

The fix presented here should be similar to only doing path folding at lower HTL, but less invasive. (I do not feel competent to estimate the effects of path folding at lower HTL - especially the longterm effects)

Longterm effects of folding at lower HTL could be that the network gets separated, and I think we don't have the math to check that. Using two buckets ensures that we still use what is proven to work (somehow) and just try to compensate for the problems due to the skewing from local requests.

4 Measuring success

To see whether the change actually works, we will use 2 metrics: The link length distribution and the success-rates per HTL.

4.1 The link length distribution

You see the current distributions in the images above. The measured distribution in the linear case (the second picture) should move much closer to the green line.

4.2 The success-rates

We should have a better success-rates at HTL bigger than 7. The paper suggests that we should have average path lengths of 13< and due to probabilistic reduction at HTL 18, going from 18 to 7 actually counts as 13 steps. In turn, there should be much less requests at lower HTL. To test this yourself, look at the statistics page in advanced mode: remote bulk fetches (CHK). To have a reference: here are the top 5 HTLs from toad, bertm and me (ArneBab).

Note also the last number in parens in each line. That's the average distance of the requested data (if I understood it correctly - if not, I'll correct that here).

4.2.1 toad

```
18  52.770% (13139,6594,37394) (.0071)
17  54.258% (13642,6261,36682) (.0064)
16  58.043% (20357,5887,45215) (.0028)
15  58.203% (17504,4020,36981) (.0023)
14  55.340% (15037,3411,33336) (.0026)
```

4.2.2 bertm

```
18 36,081% (495,1593,5787) (,0111)
17 36,671% (415,1233,4494) (,0081)
16 36,469% (801,1696,6847) (,0073)
15 31,413% (705,1394,6682) (,0059)
14 27,008% (544,1141,6239) (,0068)
```

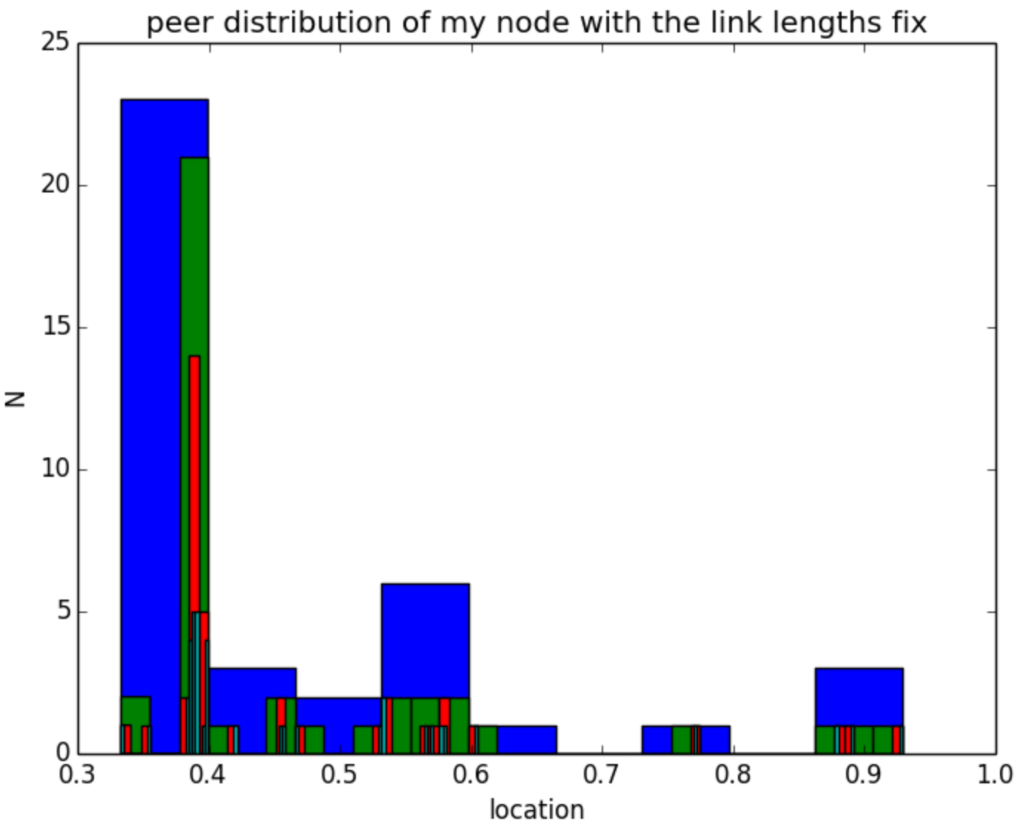
4.2.3 ArneBab

```
18 36,380% (6477,70294,211024) (,0083)
17 34,274% (4981,57742,183002) (,0073)
16 31,053% (6565,63068,224239) (,0040)
15 27,866% (4690,45558,180322) (,0029)
14 24,331% (3436,37068,166468) (,0032)
13 21,313% (2459,29873,151698) (,0037)
12 19,372% (2069,25624,142957) (,0042)
11 17,327% (1678,21957,136405) (,0043)
10 15,733% (1497,19151,131238) (,0047)
9 14,380% (1273,17219,128599) (,0048)
8 13,194% (1328,15325,126221) (,0050)
7 12,202% (1123,14012,124035) (,0051)
6 11,177% (1057,12574,121957) (,0052)
5 9,870% (915,11105,121779) (,0055)
4 8,694% (834,9832,122687) (,0057)
3 8,523% (1037,10986,141060) (,0059)
2 7,055% (1219,9823,156523) (,0062)
1 3,611% (2745,17973,573724) (,0066)
```

5 The testing build

operhiem1 deployed build 1464 which makes every opennet node report its location when asking for a connection.

toad wrote the first version of a fix which ensures that 70% of the connections have a link length shorter than 0.01. A quick plot of the locations of the peers of my node looks like this:



He then released a second build with cleaner code. In code-review we found a bug which led to high churn (connection-lifetimes of only up to 5 minutes). This is fixed in the second testing release (the connections are long-lived again).

(plotted with the script ./plot_my_node_peer_locations.py - uses pyFreenet and pylab)

To test it yourself (under GNU/Linux or OSX), just go to your freenet folder and run

```
./update.sh testing
```

If the update script fails, just try

```
bash ./update.sh testing
```

Then you can watch the success-stats and your peer distances on http://127.0.0.1:8888/stats/?fpryAdvancedMode=2.

5.1 Testing success rates

These stats are what we get when we only activate the link length distribution fix at a few nodes.

5.1.1 toad

```
18  55.988% (2417,865,5862) (.0036)
17  57.299% (2264,770,5295) (.0032)
16  64.138% (3698,884,7144) (.0017)
```

```
15  62.513% (3529,655,6693) (.0015)
14  60.268% (2985,525,5824) (.0015)
```

5.1.2 bertm

still to be added

5.1.3 ArneBab

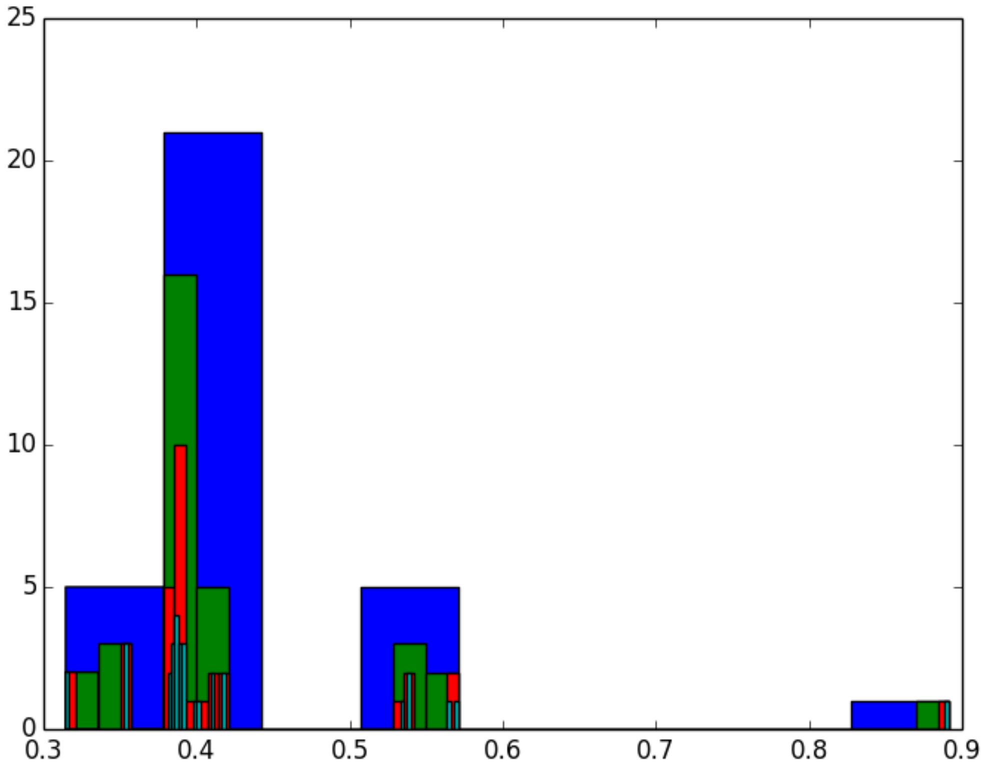
```
18 41,022% (238,1882,5168) (,0062)
17 39,749% (220,1492,4307) (,0042)
16 32,726% (190,1444,4993) (,0027)
15 30,386% (154,1231,4558) (,0021)
14 27,173% (116,1019,4177) (,0024)
13 23,874% (85,837,3862) (,0023)
12 21,802% (54,672,3330) (,0026)
11 19,944% (55,583,3199) (,0027)
10 18,678% (53,518,3057) (,0029)
9 14,420% (32,399,2989) (,0029)
8 16,336% (35,435,2877) (,0029)
7 13,272% (30,357,2916) (,0030)
6 12,266% (24,350,3049) (,0028)
5 12,048% (32,331,3013) (,0035)
4 10,373% (24,276,2892) (,0031)
3 9,957% (28,270,2993) (,0034)
2 7,818% (24,225,3185) (,0032)
1 4,878% (79,539,12670) (,0036)
```

Note: It takes much longer to connect, which is to be expected, because only 1 in 50 nodes has a location which is suitable for the 70% of short connections we want. Still this effect has to be kept in mind.

6 Deployment of the fix

The fix went mandatory at 18th August 2014.

A quick plot of the locations of my node now shows this link distribution:



Notable features might be the strong clustering around 0.3/0.4 (my locations) and the secondary cluster around 0.5/0.6 (which I do not understand).

6.1 Success rates after deployment

These success rates are what we get 2 weeks after build 1465 with the link length fix became mandatory

What we do not know is whether the results are due to the better structure or due to the leaving of many of the parasitical patched nodes with extremely many connections (see the freenet user statistics and freenet meltdown)

6.1.1 toad

```
18 45.089% (370,314,1517) (.0223)
17 50.674% (434,355,1557) (.0215)
16 50.316% (540,256,1582) (.0040)
15 54.855% (436,95,968) (.0023)
14 56.418% (472,73,966) (.0013)
13 53.673% (417,36,844) (.0010)
12 46.294% (306,25,715) (.0010)
11 43.222% (252,19,627) (.0010)
10 39.893% (201,22,559) (.0012)
9 40.681% (191,12,499) (.0011)
8 34.722% (138,12,432) (.0013)
7 30.898% (142,6,479) (.0014)
6 31.263% (142,4,467) (.0012)
5 34.753% (151,4,446) (.0014)
4 28.082% (120,3,438) (.0013)
3 26.923% (117,2,442) (.0016)
2 30.024% (119,5,413) (.0014)
```

```
1 27.932% (772,33,2882) (.0020)
```

The success-rates for toad actually decreased in the part we know (HTL >= 14), but the distances for HTL<17 decreased, suggesting better routing.

Since he used to be a seednodes it’s possible that all nodes with many peers are connected to his node. Also he has a huge store (500 GiB) and roughly 6 darknet peers at any given time.

6.1.2 bertm

still to be added

6.1.3 ArneBab

```
18 49,027% (13310,293591,625981) (,0108)
17 48,755% (12687,249795,538367) (,0078)
16 41,760% (18384,278071,709894) (,0035)
15 35,465% (14305,169416,518041) (,0012)
14 28,796% (9438,128312,478358) (,0011)
13 22,625% (5153,87315,408706) (,0011)
12 18,144% (2805,60260,347577) (,0013)
11 15,075% (1676,43048,296672) (,0014)
10 13,590% (1116,34931,265248) (,0017)
9 11,644% (855,27521,243690) (,0018)
8 10,864% (746,24038,228126) (,0020)
7 10,517% (854,21922,216567) (,0021)
6 10,679% (1016,21253,208529) (,0023)
5 8,133% (655,15129,194083) (,0023)
4 10,039% (533,19220,196768) (,0027)
3 9,120% (2379,19837,243604) (,0017)
2 7,804% (3018,20316,298987) (,0017)
1 2,372% (3663,24931,1205548) (,0023)
```

Notably the success rate for requests at hops 18 to 16 are more than 10% higher than before the fix, and at HTL 15 and 14 they are still 8% and 4% higher. This suggests that 40% more requests succeed on the first 4.5 hops (18 to 17 counts as 1.5 hops due to random decrement).

Also while the distance of the originating node (the last number in parentheses) at HTL 18 to 16 is almost unchanged, at HTL 15 and lower it drops to one third of the value before the fix which suggests much better routing: Instead of bouncing around the network at random, requests stay close to the target location. In an ideal network with 5k nodes, this would drop to roughly 0.00025 (assuming that there are 5 neighbors which all know my node: 0.0001 * 5/2).

This theoretical value is actually reached by the remote realtime fetches at HTL 15 and lower. The too high distance of the originating node suggests quite a bit of overload and backoff leading to non-optimal routing. New Load Management (NLM) might be the tool to fix that. Sadly we did not take the realtime stats before implementing the fix, so we lack some answers which could be useful. The following are my current stats:

For comparison with toad_: My store is just 600 MiB. I also have about 6 darknet peers.

```
18 51,521% (3095,53096,109064) (,0142)
17 49,619% (3653,54175,116544) (,0107)
16 72,979% (5117,38649,59971) (,0013)
15 66,579% (3981,21357,38057) (,0002)
14 54,477% (1926,13983,29203) (,0002)
13 41,023% (645,7772,20518) (,0003)
12 32,784% (273,4386,14211) (,0004)
```

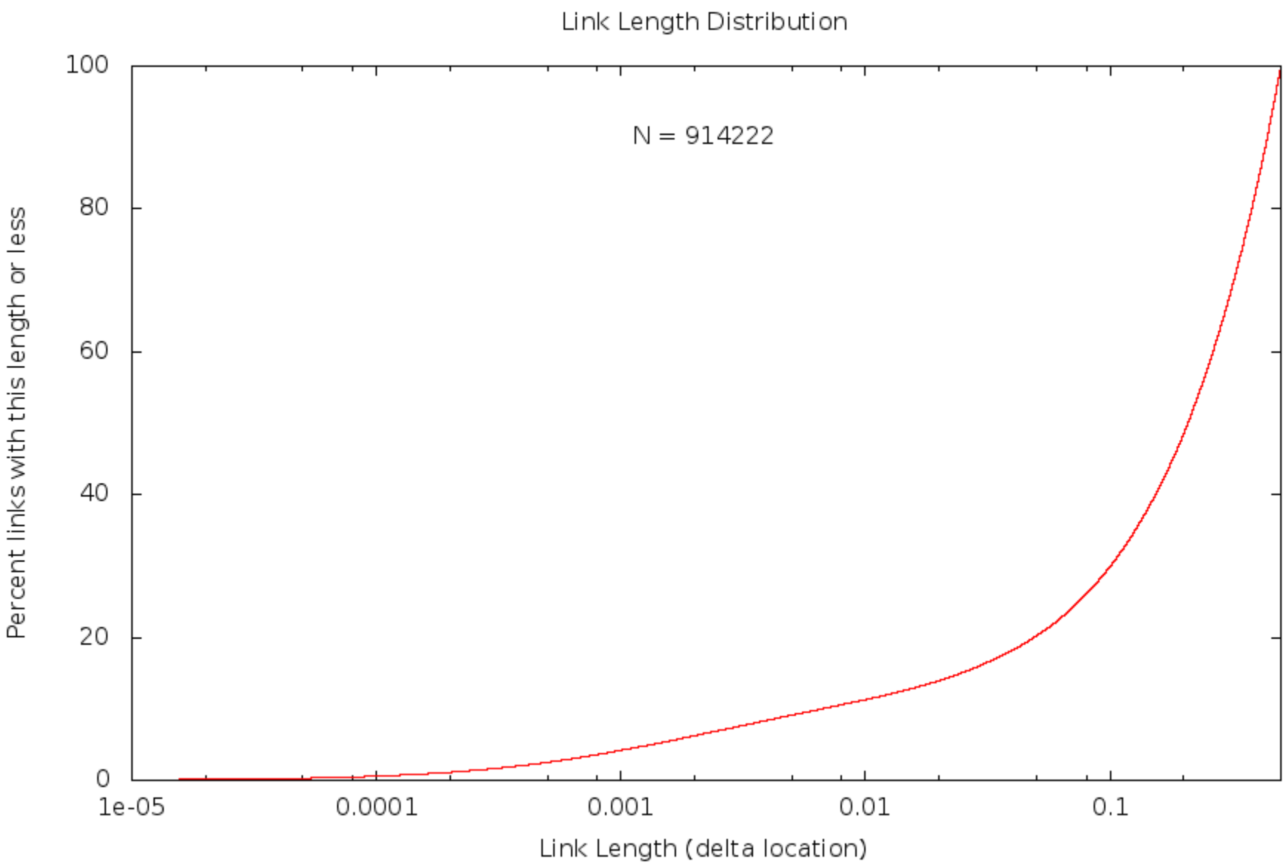
```
11 31,097% (209,3309,11313) (,0005)
10 29,163% (204,2633,9728) (,0007)
9 23,003% (129,1812,8438) (,0006)
8 22,175% (130,1803,8717) (,0006)
7 19,455% (100,1478,8111) (,0006)
6 19,089% (61,1334,7308) (,0007)
5 19,511% (74,1459,7857) (,0011)
4 21,691% (96,1502,7367) (,0008)
3 61,850% (656,23729,39426) (,0404)
2 59,674% (1653,61777,106295) (,0489)
1 15,830% (2339,9401,74163) (,0008)
```

I do not understand the high success rate at HTL 3 and HTL 2 at all. I'll ask in IRC whether someone can give answers, once this freesite is uploaded.

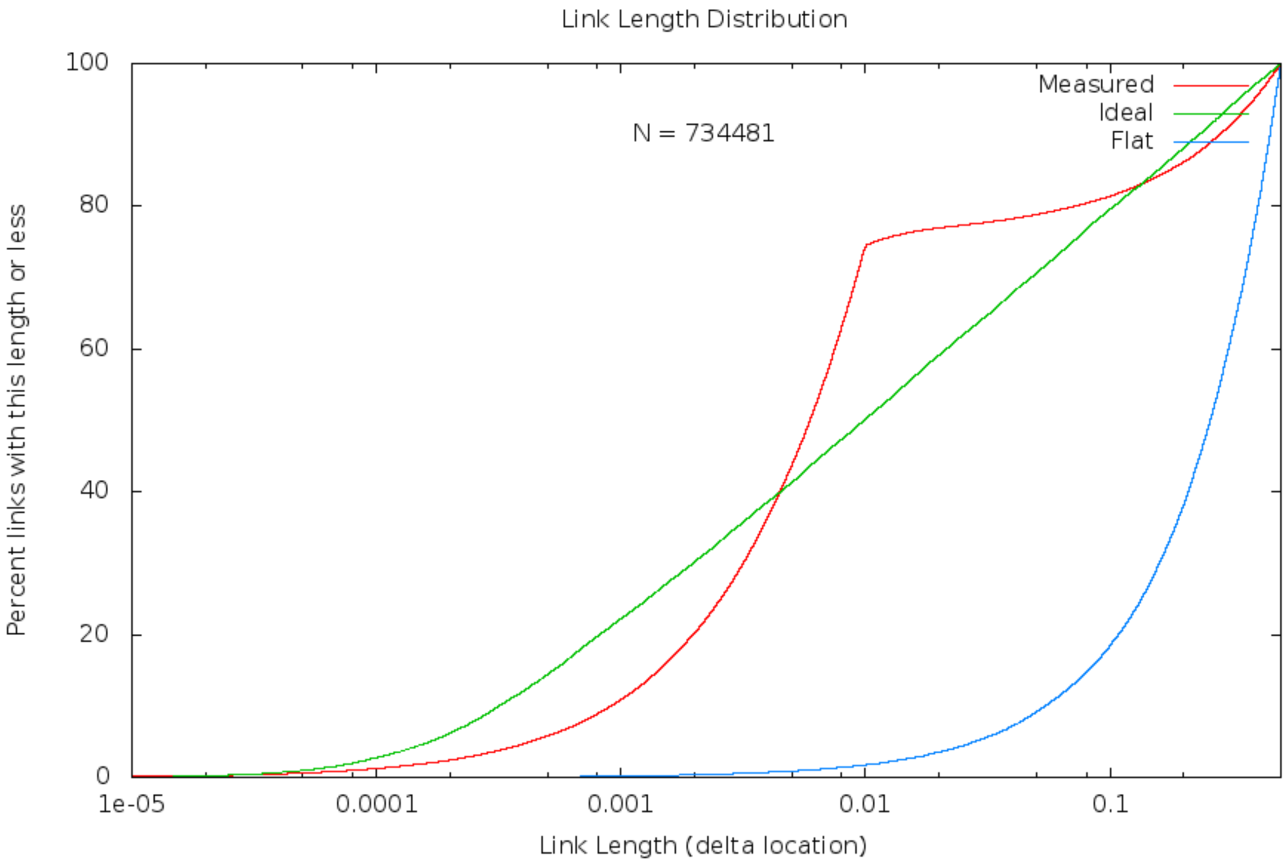
6.2 Link length distribution after deployment

Since bertm currently does not answer on IRC, the following are the old and new link length distribution from the performance-stats page:

6.2.1 old (rev 580)



6.2.2 New (rev 607)



6.2.3 Interpretation

The new line actually roughly follows the ideal line (for an 8k network). We still have a bit too few really short links, and a bit too many links at distance 0.01, but it now roughly looks as it should. The too high number of links at distance 0.01 is intentional, because that's the distance below which the network optimization seems to actually work again.

7 Please pass on this site!

There might be some disruption in the short term due to adaptions of the topography. Please pass on the link to this text, so people are informed.

Author: Arne Babenhauserheide
Created: 2014-09-23 Di 14:11
Emacs 24.3.1 (Org mode 8.2.6)
Validate